How does the precision of pseudo-counts affect learning efficiency?

Mahshid Rahmani Hanzaki [*1] **Adrian Orenstein** [*1] **Mahvash Siavashpour** [*1]

Abstract

In reinforcement learning (RL), the challenge of balancing exploration and exploitation escalates as environments grow in complexity. There is a need for exploration strategies that lead to more efficient learning and are scalable with extremely large state-action spaces. There are strong methods of exploring in MDPs with exact counts. However in MDPs with large state-action spaces, using counts becomes computationally intractable. This research investigates the impact of varying how much states are clustered to approximate state-action visitation counts. Introducing pseudo-counts an approximation over the state-action visitation counts. We hypothesise that a carefully adjusted level of state abstraction should accelerate the discovery of effective policies with minimal unnecessary state-space exploration. This research aims to identify a scalable pseudo-count model-free approach for environments with large state-action spaces.

1. Introduction

Improving the way reinforcement learning agent explores will enable agents to obtain larger returns with fewer interactions with their environment, thus being *sample efficient*. In-order to obtain a larger returns, a reinforcement learning agent must trade-off preferring actions it has found to be effective in producing reward, versus trying actions it has not selected before. This dilemma gives rise to the exploration/exploitation trade-off (March, 1991). The problem of exploration is exacerbated in environments that have large state spaces, or many actions, such as in continuous environments, where learning times can grow quite large (Kakade, 2003; Hester & Stone, 2013; Osband et al., 2016b; Ladosz et al., 2022). Environments with large state and action spaces are challenging, better methods of exploration will improve the sample efficiency of our agents in these more real-world problems.

In real-world problems with large state-action spaces, efficient exploration methods are challenging to develop but have reached super-human performance in some benchmarks (Taïga et al., 2020). A method of exploring large and complex environments in an efficient manner is highly sought after as it will better scale with compute resources (Sutton, 2019). In real-world problems, you need to spend your time effectively to learn policies that *maximise return quickly* and exploration can be very cost-prohibitave and often dangerous for the agent (Hester & Stone, 2013). Our goal in exploration is to accelerate the rate at which agents obtain larger rewards.

The myopic exploration strategies used in practice, while conceptually simple, become disproportionately expensive in large state-action environments. Noise-based methods for exploration are still used in practice due to both their simplicity and impressive performance (Mnih et al., 2013; Lillicrap et al., 2015; Fortunato et al., 2018). The consequences of this are that these methods are computationally inefficient in large state-action spaces. While exact-count based methods are expensive, recently approximations of counts (pseudo counts) have shown great success in problems with large state-action spaces (Bellemare et al., 2016; Parisi et al., 2022). However, many papers have not gone into detail on what precision of the approximation of the true count is necessary for efficient exploration. In the next section we will discuss the related work to solidify the gap in the literature we wish to pursue.

2. Background

In this section we formalise the exploration baselines we use in our evaluation section (See section 5.4). We will then introduce the RL problem framing, the Q-Learning solution method, and UCB method, upon which our algorithm is built. Later in our methods section (See section 5.3) we will formalise our methods of exploration.

A Markov Decision Process (Puterman, 1994) (MDP) can be described as a tuple of $\{S, A, P, R\}$ where S is considered the state space, A is the action space. The probability of transitioning from state s to state s' with action a is $P(s'|s, a)$. The scalar reward given at a particular timestep

[*]Equal contribution [1]The University of Alberta, Edmonton, Canada. Correspondence to: Adrian Orenstein <aorenste@ualberta.ca>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

t is R_t. In discrete environments, the size of the state-action space is finite and countable.

We define a Q function $Q(s_t, a_t)$ in terms of the policy π that takes s_t and a_t at timestep t as input. The value of the state action pair is as follows:

$$Q(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | S_t = s_t, A_t = a_t \right], \quad (1)$$

where γ is the discount factor, T is the final time step of the episode, and $\mathbb{E}_\pi [...]$ denotes the average over a finite batch of trajectories gathered using the policy π.

We use the off-policy TD control algorithm Q-Learning (Watkins & Dayan, 1992) as our policy:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)], \quad (2)$$

where α is the learning rate, and γ is the discount factor.

Upper Confidence Bound (UCB) most notably UCB-1, (Auer, 2002), is the exploration method that we intend to build upon. UCB is a model-free method that uses exact counts over the state-action pair to encourage the agent to explore less-seen state-action pairs. UCB has been traditionally applied in the context of multi-armed bandits, but more recently, there are methods that incorporate UCB in Markov decision processes. UCB uses the idea of "optimism in the face of uncertainty", where selecting state-action pairs with low counts increases the value of the Q function, causing the agent to be more optimistic, and thus the agent is more likely to pick states it has not visited before.

Defining counts on a discrete environment such as a grid world, when using a tabular learning method like Q-learning, can be done in two different fashions: state visitation counts vs state-action visitation counts. In this project, we assume we are defining our counts on state-action visitation counts to follow the original notation of the UCB-1. Aggregating counts in this setting means defining one state-action visitation count for a number of similar state-actions. Therefore, a measure of similarity among state-actions and an aggregation method is needed.

$$Q^{\text{UCB-1}}(s, a) = Q(s, a) + \kappa \sqrt{\frac{2 \log \sum_a \mathrm{n}(s, a)}{\mathrm{n}(s, a)}} \quad (3)$$

where $\mathrm{n}(s, a)$ is the exact state-action visitation count and κ trades off exploration and exploitation.

As one of our baselines we use the Boltzmann distribution sampling. The Boltzmann distribution sampling method assigns the probability p to each action based on the following softmax function:

$$p = \frac{e^{Q_t(a)}/\tau}{\sum_{b=0}^{N} e^{Q_t(b)}/\tau}, \quad (4)$$

where τ is a parameter that changes the temperature of relying on the Q-values to create the distribution, versus sampling actions more uniformly at random.

3. Related Work

This review covers the development of model-free exploration strategies in reinforcement learning, highlighting the need for computationally efficient methods of exploration in vast state-action spaces to improve the sample efficiency of our agents. We constrain our related work to focus on exploration methods in large deterministic state-action spaces. We will now cover the evolution of exploration strategies used to improve sample efficiency while managing computational constraints, with a focus on real-world applicability in environments that have large state-action spaces.

Dithering or myopic strategies for exploration become inefficient as problem spaces expand, leading to significantly larger data requirements (Osband et al., 2016a; 2019; Zhang et al., 2020). The ϵ-greedy strategy embodies the principle of balancing exploration and exploitation by occasionally forgoing the current estimate of the best action to explore a random alternative. These strategies are predicated on the assumption that over time, random exploration will accrue enough information to guide an agent towards optimal decision-making. Due to the computational demands of random exploration more sophisticated exploration techniques are needed to explore large environments efficiently.

State visitation counts used in tabular methods typically visit more novel states in a large and complex environment than a random or dithering exploration strategy (Machado, 2019). Additionally, count-based methods succeed in a variety of domains without requiring domain-specific knowledge (Sutton & Barto, 1998; Auer, 2002; Zhang et al., 2020) while having convergence guarantees (Kearns & Singh, 2002; Brafman & Tennenholtz, 2003). However, the computational requirements of exact counts grow intractable as the state-action size of the environment increases, motivating the need for approximations of the true state visitation count. This motivates two crucial considerations of our work. There is a need for both *a principled and domain-independent method of exploration*, that *visits states efficiently despite the state-action space growing large*, resulting in agents that obtain larger returns with fewer interactions with the environment.

Recent advancements have led to the development of

pseudo-counts to approximate exact counts, leading to more efficient exploration in continuous or large state-action environments (Bellemare et al., 2016). In Tang et al. (2017), states are mapped to hash codes, enabling counts based on their occurrence in a hash table. These approximations of the state-visitation count are then used to compute a reward bonus according to the classic count-based exploration theory. Tang et al. (2017) provide some surface level investigation into both the appropriate level granularity/precision of the pseudo-count, and the exploration bonus evaluated on the Atari Learning Environment (ALE) (Bellemare et al., 2012). While there have been more modern approaches using approximations of the count visitation (Abel et al., 2016; Ostrovski et al., 2017; Jin et al., 2018; Machado et al., 2020), few papers go into depth about the trade-off between granularity, and the intrinsic reward bonus hyperparameter, and how they both influence exploration efficiency. Taïga et al. (2018) is most inline with the intention of our work, as they study the granularity of their model-based approximation for the count and the reward bonus parameter. However very little work has been done to understand model-free approaches, and this paper seeks to bridge this gap in knowledge.

4. Research question

Our research question is *How does changing the precision of pseudo-counts affect how quickly our agents learn good policies?* Pseudo-counts are a method of aggregating the state, so that if counts are too sparse in a large and complex environment, we can decrease the granularity or sparsity of our count including more states in our pseudo-count (Li et al., 2006). Our expectation is that by decreasing the granularity of our pseudo-count, our state visitations effectively "cover more of the state space", and thus our agent should more quickly explore further out into its state space during training. Our hypothesis is that for a particular environment, there is likely a good configuration of the precision/granularity of state abstraction where we will see better sample efficiency for particular settings over others. We expect to find a granularity or precision setting for our pseudo-count that will result in our agent accumulating more reward much earlier in training.

5. Experiment Design

In our experiment design section, we establish the environment, metrics, baselines, and our method. We hypothesise that the resolution of our state aggregation will have an effect on how quickly the learning algorithm finds a better policy.

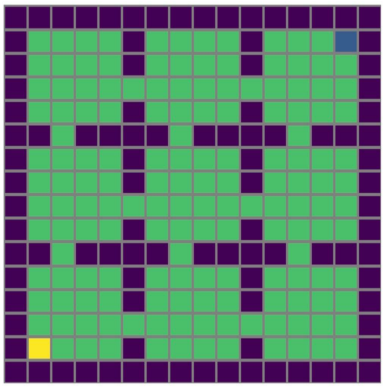


Figure 1: The 9-room environment. The agent starts from bottom left corner and needs to navigate through the green tiles to reach the goal in to top right corner.

5.1. Environment

We choose the 9-room environment from Taïga et al. (2018) (see Figure 1) as it will enable us to try various intuitive ways to aggregate the state space and answer our research question.

In this environment, the agent starts from the bottom left (yellow tile) and needs to reach the top right state (blue tile). Each room consists of 4×4 grids. The agent has access to four discrete actions: *up, down, left, right*. For each action that does not lead the agent to a goal state, the agent receives a reward of 0. Transitions are deterministic, the agent always moves to an open tile, and moving into walls does not move the agent. The environment runs until the agent reaches the goal, at which point the agent is rewarded with a positive reward of 1, and the episode starts over from the initial position (Taïga et al., 2018).

The 9-room environment is sufficiently large to provide rooms that are not along the optimal path to the goal, and the goal state is initially unknown to the agent. As the 9-room environment contains many tiles that are not the objective, we can aggregate the tiles in each room to answer our research question. Our rationale for selecting the 9-domain environment is that it means that we can focus on the influence of various methods of aggregating states rather than requiring the agent to learn them itself.

5.2. Metrics

In order to assess our exploration algorithms, we define a few quantitative and qualitative ways of analysing the behaviour of our exploration algorithms.

Quantitatively, we employ two different metrics: discounted return and the number of episodes to reach the maximum performance of the learning method. The discounted return is the weighted sum of rewards during an episode which is

defined as,

$$G_e = \sum_{t=1}^{T_e} \gamma^{t-1} R_t, \tag{5}$$

where γ is the discount factor, R_t is the reward received at time step t, e is the current episode number and T_e is the terminal time step in the current episode. We also use the number of episodes it takes for the agent to reach the highest performance as a measure of sample efficiency of learning when combined with each of the exploration algorithms. This metric is calculated by the number of episodes it takes for the agent to converge to a fixed discounted return.

We also want a measure of how many steps did it take for the learning method to plateau in performance, we call this point the timestep the agent has converged. The condition for a converged algorithm is when it no longer improves in discounted return within some θ value. Using an arbitrary window of size w episodes, we can compute the average over this window and subtract the current episode's return to determine whether we have improved our policy beyond a θ amount:

$$\left| \left(\frac{1}{w} \sum_{i=N-w}^{N-1} G_i \right) - G_N \right| \leq \theta, \tag{6}$$

where N indicates the number of the episode, G is the discounted return, and θ is the tolerance for determining whether our policy has improved. We are using a sliding window of w episodes to compare their average performance against the current episode. We report N^{th} as the episode on which the agent converges. In our experiments, the tolerance $\theta = 3e^{-4}$ and window size $w = 5$.

We also want a qualitative measure to compare the learning algorithms. We use a heatmap of the agent's state visitation counts averaged across 50 runs. We separate the 200 episodes into two sections, before convergence and after convergence, using Equation 6, and plot each as a different heatmaps. The first heatmap can show how each of the algorithms explores in the environment, and the second one depicts the converged visitation map.

5.3. Our method

In this section we introduce our learning algorithm that uses pseudo-counts by aggregating the states. We use the Q-Learning algorithm formalised in Equation 2, using the UCB formulation in Equation 3, and in this section we discuss how we extend UCB to use pseudo-counts. We will now define how we will aggregate states.

Aggregating States In this work, we are focused on understanding how changing the precision of our state aggregations affects the learning efficiency of our agent and there are challenges in defining a state-aggregation method. While learning a useful mapping of states to aggregated state is a large open body of work, we will assume that the agent receives aggregations and does not learn them. We opt to use human-defined aggregated states and leave learning the state aggregations for future work (see Section 9). In our 9-room environment, intuitively, we can break each room into quadrants, or use the whole room as a state-aggregation. Figure 2 illustrates each level of these aggregations using 1x1 (exact counts), 2x2 dividing the room into quadrants and 4x4 which aggregates the whole room.

Incorporating the bonus reward In this setting, whenever the agent visits any state-action pairs in the aggregated tiles, the count of the corresponding state-action aggregation increases. To implement our bonus as an intrinsic reward, we simply change Equation 2 to use the task reward R and our count-based intrinsic reward. This bonus reward formulation R^{bonus} helps encourage the agent to explore less visited states, defined as:

$$R^{\text{bonus}} = R + \kappa \sqrt{\frac{2 \log \sum_a \text{n}(s,a)}{\text{n}(s,a)}}, \tag{7}$$

where R is the reward received from the environment, $\text{n}(s,a)$ is the state-action visitation count, and κ is a scaling coefficient. For now, we assume we keep count of the visited aggregated states in each run, where the number of aggregated states can be 1 when referring to exact visitation counts.

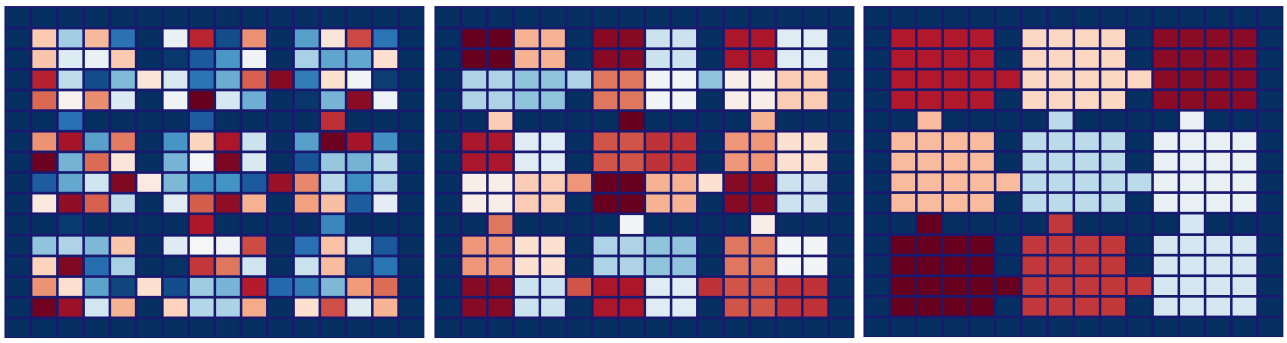
The Q-Learning algorithm can then be modified as follows:

$$Q(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R^{\text{bonus}}_{t+k+1} | S_t = s_t, A_t = a_t \right]. \tag{8}$$

5.4. Baselines

Inorder to put our results into perspective, we compare our count and pseudo-count methods above to two baselines. The non-count based methods we intend to use are ϵ-greedy and weighted sampling for exploration (softmax exploration). The ϵ-greedy policy is commonly used for exploration in complex environments (Machado, 2019) and weighted sampling (or softmax sampling), using the Boltzmann distribution (Sutton & Barto, 1998), will provide a baseline using a less random exploration strategy.

To use counts for exploration we assume that agent is acting ϵ-greedy w.r.t. the Q function. Acting greedy when using a positive intrinsic reward term in an environment when the values of the Q function are initialised to zero, and reward received from the environment is predominantly zero leads to positive values of Q function on each time step. This means the agent is rewarded positively simply by taking an action proportional to each state-action visitation count. Without another measure of exploration the agent would



(a) 1x1 Aggregation (Exact counts) (b) 2x2 Aggregation (c) 4x4 Aggregation

Figure 2: Different aggregations used for evaluation

take the same action as it yields a positive reward. Therefore, employing ϵ-greedy aids the agent in taking different actions, promoting exploration of other state-action pairs.

5.5. Hyperparameters

We compare our baselines when acting ϵ-greedy w.r.t the Q function to maximize the discounted return. Each experiment is conducted over 200 episodes and the results of each episode is averaged over 50 runs and 50 seeds. A constant discount factor $\gamma = 0.99$ is used across all experiments. For other parameters, namely α, ϵ, κ, and τ, we conduct a sweep over various values to identify a set of parameters that yield the best performance for each exploration algorithm. The parameter sweeps include $\alpha \in \{0.01, 0.1, 0.2, 0.5\}$, $\epsilon \in \{0.01, 0.1\}$, $\tau \in \{0.01, 0.1, 1, 10\}$, and $\kappa \in \{0.001, 0.01, 0.05, 0.1\}$. Through a search on all combinations of these hyperparameters for each algorithm, we compare the sum of discounted return on all episodes and select the set of parameters that result in the highest performance. The best set of hyperparameters for each algorithm is reported in Table 1.

Table 1: Parameters that led to the best performance of each algorithm

ALGORITHM	α	ϵ	κ	τ
ϵ-GREEDY	0.01	0.01	-	-
SOFTMAX SAMPLING	0.2	-	-	0.01
UCB EXACT COUNTS	0.2	0.01	0.05	-
UCB AGGREGATION 2×2	0.5	0.01	0.05	-
UCB AGGREGATION 4×4	0.5	0.01	0.001	-

To ensure the reproductability the experiments were done over different random seeds where $seed \in \{0, 1, ..., 49\}$.

6. Results

We compared the performance of each of our proposed exploration algorithms against the ϵ-greedy and softmax exploration. The hyperparameters used for each algorithm is chosen from Table 1. Figure 3 depicts the discounted return across 200 episodes averaged over 50 runs. All algorithms were able to converge during these episodes.

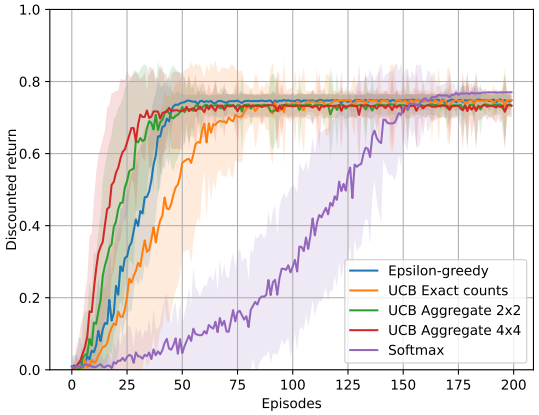


Figure 3: **Discounted return for each episode** averaged over 50 runs. Each curve denotes one of the exploration algorithms used with Q-Learning. The algorithms plotted are: Q-Learning with ϵ-greedy (blue), softmax sampling (purple), exact counts (orange), 2×2 state aggregation for counts (green), and 4×4 state aggregation for count (red).

7. Discussion

By comparing the algorithms in Figure 3, we observe that 4×4 and 2×2 aggregations exhibit the fastest learning and converge more quickly than the other algorithms, closely followed by ϵ-greedy.

When compared to ϵ-greedy and pseudo-counts, exact counts seem to perform worse when it comes to how fast they reach the highest performance. This shows that using exact visitation counts in updating the Q function will encourage too much exploration in the beginning which will in turn affect the sample efficiency of the learning

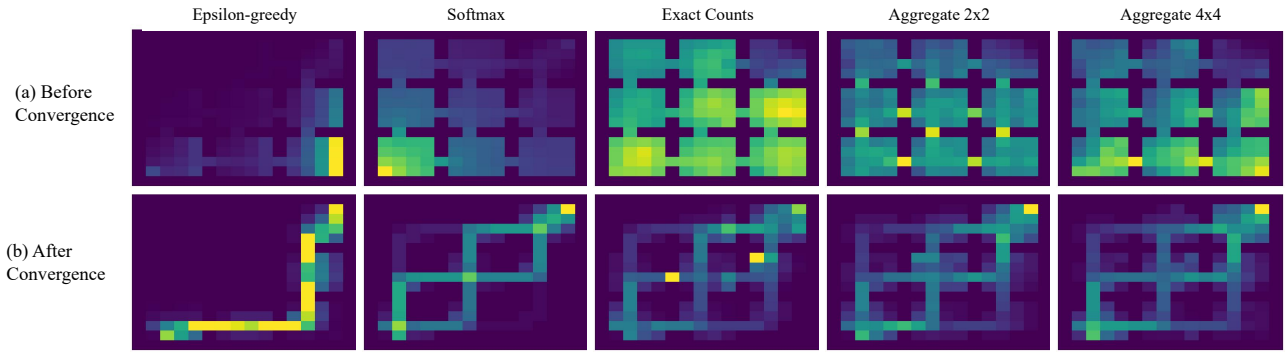


Figure 4: **Sum of state visitations** of different algorithms in episodes (a) before convergence and (b) after convergence. The convergence episode of each of the algorithms is calculated using Formula 6.

algorithm. Furthermore, by comparing exact counts with pseudo-counts, the process of aggregation appears to have reduced variance in the early stages of learning.

In exact counts, every state-action pair has a different visitation count which makes every less visited state very appealing to the agent. Meaning even if states with different visitation counts are neighbours and have same priority in terms of reaching the goal, the agent is encouraged to visit them all. The agent will start to exploit once the auxiliary term in reward becomes small and converges to zero which is towards the final episodes. However, once converged, it is able to perform better than the pseudo-count algorithms. This is also due to the fact that exact count is more precise in terms of exploring each and every state of the environment for finding the best path to the goal. On the other hand, aggregation methods treat similar states as equal in terms of visitations, allowing the agent to do a generalisation over state-space and find the best path to the goal faster.

7.1. Heatmap and convergence

Figure 4 illustrates the average state visitation counts before and after convergence. Before convergence, ϵ-greedy explores too little of the upper rooms, and Softmax explores too much in the first room. Exact count explores all rooms almost equally, whereas the pseudo-count methods explore rooms less due to doorways being highlighted higher relative to rooms. Exact count methods seem to explore the space more evenly, and aggregating the state pushes the agent to explore states further away from the starting goal.

Interestingly, ϵ-greedy and count based methods, when converged, reach a lower discounted return compared softmax. This phenomenon could be mitigated with a schedule to decrease exploration of the count-based methods and ϵ-greedy methods over time.

In the 9-room environment, there are 6 equally optimal paths. ϵ-greedy finds one of the optimal paths. Softmax appears to prefer 4 of the paths, and not the other 2 on the top left and bottom right. Whereas exact counts and the 2×2 and 4×4 methods gradually weight them equally. Exact counts and pseudo-counts visit more optimal paths in the 9-room environment than ϵ-greedy and softmax, indicating count-based methods have converged to the correct Q-value for each path.

8. Conclusion

In this work we investigate how using pseudo-counts as a heuristic for exploration affects the sample efficiency learning. To test our hypothesis, we manually defined three different levels of state-action aggregation on each of the rooms in the grid by combining tiles which were close. We incorporate visitation counts into the tabular Q-Learning method by introducing an intrinsic reward that is inspired by UCB. The results show that the pseudo-count methods were able to learn faster than all other methods with 4x4 aggregation outperforming the others. Exact counts, on the other hand, learn slower than pseudo-counts and ϵ-greedy due to continually pushing the learning algorithm to revisit lesser explored state-action pairs. We conclude that, yes, the precision does indeed affect the learning efficiency of our algorithms.

9. Future work

While we tried aggregating quadrants of a room, and whole rooms, we could try aggregating multiple rooms together to encourage the agent to move faster toward the top right corner. We would expect this to further increase the rate at which the learning algorithm converges to the converged policy.

Our implementation of the Q-Learning method uses the true state as input but uses the aggregated state for the intrinsic reward bonus. However, should the policy and Q function receive as input the state or aggregated state? We decided that the policy should use state information, and not aggregated state, however this affects generalisation and

may need to change when moving to problems with larger state-action spaces.

The bonus could either be added to the reward (as we did, creating an intrinsic reward), added to the value as in UCB, the policy to select an action, or as input as a feature to the policy. It remains an open question to determine which one of these four methods is most effective.

The limitation of this work is that we do not evaluate how our pseudo-count methods of exploration performs as the state-action space increases. This is an important extension to this work as pseudo-counts are necessary for large state-action environments, whereas our 9-room example contains relatively few states so that we can compare against exact count methods. Either we can procedurally generate larger mazes, or more tiles within each of the 9-rooms, or we can move onto a problem with large state spaces like the inverted pendulum gym environment (Brockman et al., 2016).

We have provided a method of using pseudo-counts for exploration. There are methods of performing deep exploration using exact counts, namely Parisi et al. (2022). However, methods of deep exploration using counts have not yet been combined with pseudo-counts. We expect that by learning a method of performing state-aggregation and combining the pseudo-count based method with deep exploration will further improve the efficiency of the exploration strategy.

Contributions

1. Abstract, Introduction and Related work (Adrian wrote, the rest edited).
2. Research Question and Background (Adrian formulated, Mahshid wrote, Adrian and Mahvash edited).
3. Experimental Design (Adrian formulated, Mahvash wrote, Adrian and Mahshid edited).
4. Methods (Adrian formulated, Mahshid and Mahvash wrote, Adrian edited).
5. Results (Mahshid wrote, Adrian and Mahvash edited).
6. Discussion (Mahvash wrote, Adrian and Mahshid edited).
7. Conclusion (Mahvash wrote, Adrian and Mahshid edited).
8. Future work (Adrian wrote, the rest edited)

References

Abel, D., Agarwal, A., Diaz, F., Krishnamurthy, A., and Schapire, R. E. Exploratory gradient boosting for reinforcement learning in complex domains. *CoRR*, abs/1603.04119, 2016. URL http://arxiv.org/abs/1603.04119.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov): 397–422, 2002.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012. URL http://arxiv.org/abs/1207.4708.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *CoRR*, abs/1606.01868, 2016. URL http://arxiv.org/abs/1606.01868.

Brafman, R. I. and Tennenholtz, M. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3(null):213–231, mar 2003. ISSN 1532-4435. doi: 10.1162/153244303765208377. URL https://doi.org/10.1162/153244303765208377.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. 2018. URL https://openreview.net/pdf?id=rywHCPkAW.

Hester, T. and Stone, P. Texplore: real-time sample-efficient reinforcement learning for robots. *Machine Learning*, 90(3): 385–429, 2013. doi: 10.1007/s10994-012-5322-7. URL https://doi.org/10.1007/s10994-012-5322-7.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is q-learning provably efficient? *CoRR*, abs/1807.03765, 2018. URL http://arxiv.org/abs/1807.03765.

Kakade, S. M. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Kearns, M. and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49:209–232, 2002. URL https://link.springer.com/article/10.1023/A:1017984413808.

Ladosz, P., Weng, L., Kim, M., and Oh, H. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, sep 2022. doi: 10.1016/j.inffus.2022.03.003. URL https://doi.org/10.1016%2Fj.inffus.2022.03.003.

Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for mdps. 2006.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

Machado, M. C. *Efficient Exploration in Reinforcement Learning through Time-Based Representations*. PhD thesis, University of Alberta, 2019.

Machado, M. C., Bellemare, M. G., and Bowling, M. Count-based exploration with the successor representation. *CoRR*, abs/1807.11622, 2020. URL http://arxiv.org/abs/1807.11622.

March, J. G. Exploration and exploitation in organizational learning. *Organization science*, 2(1):71–87, 1991. URL https://sjbae.pbworks.com/f/march+1991.pdf.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. Deep exploration via bootstrapped DQN. *CoRR*, abs/1602.04621, 2016a. URL http://arxiv.org/abs/1602.04621.

Osband, I., Van Roy, B., and Wen, Z. Generalization and exploration via randomized value functions. pp. 2377–2386, 2016b.

Osband, I., Roy, B. V., Russo, D., and Wen, Z. Deep exploration via randomized value functions, 2019.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Parisi, S., Tateo, D., Hensel, M., D'Eramo, C., Peters, J., and Pajarinen, J. Long-term visitation value for deep exploration in sparse reward reinforcement learning. *Algorithms*, 15(3):81, 2022. URL http://arxiv.org/abs/2001.00119.

Puterman, M. L. Markov decision processes: Discrete stochastic dynamic programming. 1994. URL https://api.semanticscholar.org/CorpusID:122678161.

Sutton, R. S. The Bitter Lesson, March 2019. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html. [Online; accessed 16. Oct. 2023].

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6. URL https://www.worldcat.org/oclc/37293240. tex.bibsource: dblp computer science bibliography, https://dblp.org tex.date-added: 2023-10-16 12:21:05 -0600 tex.date-modified: 2023-10-16 12:21:12 -0600 tex.timestamp: Fri, 17 Jul 2020 16:12:40 +0200.

Taïga, A. A., Courville, A., and Bellemare, M. G. Approximate exploration through state abstraction. *arXiv preprint arXiv:1808.09819*, 2018.

Taïga, A. A., Fedus, W., Machado, M. C., Courville, A. C., and Bellemare, M. G. On bonus based exploration methods in the arcade learning environment. 2020. URL https://api.semanticscholar.org/CorpusID:260632931.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. A study of count-based exploration for deep reinforcement learning. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA*, pp. 4–9, 2017.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8:279–292, 1992.

Zhang, Z., Ji, X., and Du, S. S. Is reinforcement learning more difficult than bandits? A near-optimal algorithm escaping the curse of horizon. *CoRR*, abs/2009.13503, 2020. URL https://arxiv.org/abs/2009.13503.